Demonstrating Programs against Adversaries

AUTHOR(S):

SAKURAI, Kouichi; IWAMA, Kazuo

数理解析研究所講究録
906巻 1995年 170-177

Demonstrating Programs against Adversaries*

1st of March, 1995

櫻井 幸一 *Kouichi SAKURAI*
岩間 一雄 *Kazuo IWAMA*

九州大学 工学部 情報工学科
Department of Computer Science and Communication Engineering,
Kyushu University, Hakozaki, Higashi-ku, Fukuoka 812-81, Japan
{sakurai, iwama}@csce.kyushu-u.ac.jp

Abstract

Methods of demonstrating correctness of programs without revealing any computing process nor computed results are investigated. A protocol to demonstrate the program to answer whether two given graphs are isomorphic or not, which is secure against adversaries, is presented. Also, a theoretical upper bound on the class of problems having such protocols is given, which suggests the guaranteed security could lower the power of the demonstration systems. The obtained results are based on no assumptions such as the intractability of factoring or the existence of one-way functions.

Key Words

Zero-Knowledge Proofs, Program Checking, Graph Isomorphsim Problem, Bitcommitment scheme, Interactive Protocol, Computational Complexity

1 Introduction

1.1 Our results

Suppose you have discovered an efficient algorithm to solve the graph isomorphism problem. The graph isomorphism problem is to decide whether two given graphs are isomorphic or not., i.e. whether there is a bijective mapping (a permutation) from the nodes of one graph to the nodes of the second graph such that the edge connections are preserved. Until today, this problem is still unsolved in the sense that no efficient algorithm for it has yet been found. So, you would strongly like to announce that you have found a new algorithm for the problem by *demonstrating* the algorithm (the program). However, you must do the demonstration carefully so as to reveal as little information as possible in order to avoid the verifier's getting important information (e.g. the algorithm, or solutions of instances etc.) via your demonstration.

For this end, one could try to use the zero-knowledge interactive proof system for membership of languages (is input x a member of language L ?) proposed by Goldwasser et al. [GMR85]. In particular, Goldreich et al. [GMW86] proposed a protocol to show that two given graphs are isomorphic without revealing an isomorphic permutation, and also presented a different protocol to show that two graphs are not isomorphic without giving any additional information. Let us consider the following simple protocol obtained from these two GMW-protocols above. If the given two graphs are isomorphic, then the prover shows the fact to the verifier by the first zero-knowledge protocol for graph isomorphism. Otherwise, when the two graphs are not isomorphic, then the prover executes the second zero-knowledge protocol for graph non-isomorphism. Thus you can convince the verifier

*A detailed manuscript is available from the first author.

that your program correctly works for any two graphs, isomorphic or nonisomorphic. Unfortunately, however, the protocol above leaks the one-bit information that these two graphs are isomorphic or nonisomorphic, which could be a very important information for adversaries. Namely, the method does not give a perfect answer to your demand.

This paper designs a new protocol and shows the first theorem.

> **Theorem I:** *Under no unproven complexity assumptions, the correctness of the program to solve the graph isomorphism decision problem is interactively demonstrated without revealing any information (even against computational unlimited powerful verifiers).*

This paper also explores what programs can be demonstrated via zero-knowledge fashion, and gives the following.

> **Theorem II:** *If the correctness of the program to a function F is interactively demonstrated without revealing any information, then the bounded error probabilistic polynomial-time algorithm with an oracle to compute the function F is no more powerful than the bounded error probabilistic polynomial-time algorithm with an* NP*-oracle, i.e. $BPP^F \subseteq BPP^{NP}$.*

This result contrasts with the result [LFKN90] that we can demonstrate, if not taking account of security, the *powerful* program to compute #P-complete functions (e.g. computing the permanent of 0-1 matrix), and suggests the guaranteed security could lower the power of the demonstration systems.

We do not know if this characterization is tight. For example, it is open whether this kind of secure demonstration of the program is possible for solving NP-complete problems. This question is related to the important open problem that NP-complete problems have checkers [BK89].

1.2 Previous results

1.2.1 As for the 1st theorem

Galil et al. [GHY85] investigated minimum-knowledge interactive proofs for decision problems, in which the prover tries to inform the verifier the value of functions (particularly 0 or 1 of a boolean predicate) with revealing nothing to the *third eavesdropper*. Furthermore, Impagliazzo et al. [IY87] presented a general method for minimum-knowledge proofs of any computation. We should note that, in their minimum-knowledge proofs, the verifier obtains the result of the prover's computation (e.g. the value of a boolean predicate).

Feige, Fiat, and Shamir [FSS87] initiated the study of protocols which achieve further security such that even the verifier can get no computation results nor any information. They [FSS87] showed, under the assumption that secure public key encryption schemes exist, the prover can show that a statement on a conjecture is true without telling anything new about the conjecture (not even whether the prover found a proof or a counterexample). However, their result can only be applied to the problems that belong to NP$\cap$ co-NP.

Though we can extend the idea in [FSS87] into all languages having interactive proofs by using the result in [BGG$^+$88], in their argument of [BGG$^+$88] the prover is not practical, i.e. the prover needs much more power than solving the given problem even though the problem is not so difficult. Thus, the discussion above fails to meet the original goal of [FSS87], in which the prover requires only (minimum) knowledge to solve the problem. In addition, the one-way functions are required as a fundamental tool to hide information, hence the achieved security is guaranteed against only computationally bounded adversaries. Thus, it has been open to find a method of demonstrating a program to solve the graph isomorphism against the unlimited powerful adversary.

Moreover, to emphasis originality of our proposed scheme, we must compare our scheme with Yung's method [Yun89]. Yung [Yun89] formulated a notion of interactive proofs of computational

power, which is very similar to our scenario, and proposed a way in which one party can prove possession of certain computational power without disclosing any algorithmic detail about this computational task. Yung's idea uses zero-knowledge proofs of knowledge [FSS87, TW87] as subprotocols and the outline is that (1) first the verifier randomly selects an instance with a solution, then sends the instance with a zero-knowledge proof that the verifier knows a solution of the instance, and next (2) the prover, after solving the problem on the instance, shows via a zero-knowledge manner that the prover knows one of the solutions for the instance. Then the problems to which Yung's method can apply is restricted within (probabilistic polynomial-time) "samplable" and "verifiable" problems, and Yung's approach seems hard to extend into computational power to solve problems beyond NP. Taking the graph isomorphism problem, we consider this point. Indeed we may apply Yung's method above into the computational power to solve the graph isomorphism problem, however, in this case we fail to discuss formally that the prover has a power to judge that a given pair of graphs are *non*-isomorphic rather than computing an isomorphic permutation between an isomorphic pair of graphs. On the other hand, in our scheme we can directly deal with such a power and give the strict proof. Thus, our scheme sugguests zero-knowledge proofs of ability to solve problems beyond NP.

1.2.2 As for the 2nd theorem

The computational complexity of zero-knowledge proofs without any unproven assumption was initially investigated by Fortnow [For87], then followed by Aiello and Hastad [AH87]. They [For87, AH87] showed an upper bound that languages having perfect zero-knowledge proofs for membership must lie in AM $\cap$ co-AM. Combined with the fact that PSPACE-complete languages have interactive proofs for membership [Sha90], their upper bound [For87, AH87] indicates the zero-knowledgeness could lower the power of the systems. We must note that the discussion in [For87, AH87] is heavily depended upon the conditions of GMR-setting, wherein even the powerful prover cannot convince the verifier to accept in the case when $x \notin L$.

On the other hand, in our systems wherein the prover demonstrates the program to decide membership of L, the verifier accepts not only for $x \in L$ but also for $x \notin L$, unless the prover's program is incomplete. Therefore, we cannot apply directly the argument in [For87, AH87] into our situation. Note that, if we do not consider security of protocols, we can demonstrate very powerful programs (e.g. computing the permanent of a given 0-1 matrix) as shown by [LFKN90]. Thus, the 2nd theorem can be viewed as another evidence that the power of proof systems could be restricted by imposed securities.

1.3 Basic Idea and Techniques Used

1.3.1 Regarding the 1st theorem

The prover have to show two opposite statements that one given pair of graphs are isomorphic and that another given pair are non-isomorphic in the same manner. Otherwise, one-bit information which a given pair of graphs belongs to GI or GNI is released. So, instead of interactive proof systems in the original sense of [GMR85], we watch weak proof systems so called "arguments" (in other words, computationally sound interactive proofs) [BCC88]. Arguments avoid the prover's cheating by assuming that the prover cannot perform some cryptographic task. Consequently arguments are not proof systems in the sense of [GMR85], becuase in arguments for menbership of a language L, a powerful prover can cheat the verifier even for a given $x \notin L$ if he has enough power to break the cryptographic assumption.

Thus, a powerful prover convinces the verifier not only for L but also for the $\overline{L}$. Our basic idea is to investigate such a cheating prover's exact power of executing the cryptographic task, and we regard the conversation between the cheating prover and the verifier as a kind of proofs for the complement of the given language. The technical outline of the proposed method is that we transform the (zero-

knowledge) proof for the language of graph isomorphism [GMW86] into a protocol so that provers can convince the verifier to accept for non-isomorphism graphs *with preserving the propoerty that the prover cannot convince the verifier without possessing an isomorphism permutation of input graphs.*.

Fo this goal, we develop a new bit-commitment scheme. This bit-commitment scheme is constructed based on the input pair of graphs, and consists of two phases: a commitment phase and a recovering phase, in which two persons sender and receiver communicate. In the commitment phase the sender, on input a random bit b to be committed and auxiliary pair of two graphs (G_0, G_1), computes in expected polynomial time *with an oracle to decide the membership of GI* a commitment key *com*, then sends (G_0, G_1, com) to the receiver. The pair (G_0, G_1, com) has the properties: (A) in both cases when $G_0 \cong G_1$ and when $G_0 \ncong G_1$, it is possible for the expected polynomial time sender who has an oracle to decide GI to reveal evidence of both $b = 0$ and $b = 1$ for the commitment key. (B) when $G_0 \cong G_1$, any sender cannot open both $b = 0$ and $b = 1$ for a committed key without possessing an isomorphism permutation between G_0 and G_1. As a consequence of the property (A), in this bitcommitment scheme, no adversary can guess the committed bit b significantly better than guessing at random, even though adversaries are infinitely powerful.

The full protocol consists of the commitment scheme above and the parallel execution of the original ZK-proof for graph isomorphism [GMW86]. Recall the parallel execution of the GMW graph-isomorphism proof, in which the prover sends k-tuple graphs $H = (H_1, \ldots, H_k)$ then the verifier sends $b = (b_1, \ldots, b_k)$ to ask the prover isomorphisms between H_i and G_{b_i} for $i = 1, \ldots, k$. In our proposed protocol, the prover commits each bit of the first message H of the GMW-proof, and instead of sending directly the graphs H, the prover sends the verifier the committed keys of H. For the verifier's challenges $b = (b_1, \ldots, b_k)$, the prover sends k-tuple isomorphism permutations with decommitted information of these committed keys.

To the readers familiar with the Feige-Shamir's 4-move perfect zero-knowledge argument for Hamiltonian cycle [FS89]: Our proposed protocol is regarded as a variant of the Feige-Shamir's 4-move scheme for Hamiltonian cycle [FS89] which is shown to be a perfect zero-knowledge argument. More precisely,

1. We apply the Feige-Shamir's protocol into another NP-problem, graph isomorphism instead of Hamiltonian cycle.

2. We modify the Feige-Shamir's developed bitcommitment scheme consists of a special discrete logarithm problem into a bitcommitmemt scheme constructed from a pair of graphs which is the common input of the prover and the verifier.

As the original FS-scheme, a powerful cheating prover convince the verifier when the input graphs are not isomorphic, however, our novel approach is to analyze the exact power of such a cheating prover and formally show that this cheating requires the power to decide the input pair of graphs are non-isomorphic.

1.3.2 Regarding the 2nd theorem

Instead of applying the discussion of Fortnow et al. [For87, AH87], we adopt an alternative approach by Bellare and Petrank [BP92] which is to measure how much power is sufficient for the prover to execute perfect zero-knowledge protocol. They showed that any perfect zero-knowledge interactive proofs for membership of languages have (statistical) zero-knowledge interactive proofs for membership of language with probabilistic polynomial-time NP-oracle prover, and obtained an upper bound such that $\mathsf{PZKIP} \subseteq \mathrm{BPP}^{\mathrm{NP}}$. The upper bound $\mathsf{PZKIP} \subseteq \mathrm{BPP}^{\mathrm{NP}}$ by Bellare and Petrank [BP92] is weaker than the previous one [For87, AH87], however, Bellare and Petrank's argument can be applied to our model because the apparent messages exchanged in the transformed protocol is not modified in Bellare and Petrank's argument. Thus, our model of demonstrating system clarifies a merit of the argument by Bellare-Petrank on the proving power of zero-knowledge proof system.

1.4 Overview of this paper

The next section gives the definition of our model on demonsrating programs with remaking previous definitons of interactive proofs and program checkers. The proofs of the main theorems are omitted from this abstract. Final section mentions some open problems.

2 Defining our model

2.1 Demonstrating the correctness of programs

This paper considers "interactive demonstration of programs" in which a (probabilistic polynomial-time) demonstrator, who has a program to compute a given function, tries to convince that the prover's program correctly works to (probabilistic polynomial-time) verifier by interaction for a polynomial number of rounds with polynomial-sized messages.

Such a variant of interactive proofs is introduced by Bellare and Goldwasser [BeGS94], wherein the prover is not unlimited power and must run in probabilistic polynomial time given access to the given language L as an oracle, in order to clarify how the power of the prover effects the power of proof systems. This part is the same setting as ours, however, as the original interactive proof systems of GMR, the goal of the prover in their model is to show the verifier that a common input x belongs to a given language L, and in the case of $x \notin L$ no prover cannot convince the verifier to accept. On the other hand, in our model, the prover show not only the fact that $x \in L$ but also $x \notin L$, so the prover tries to convince the verifier to accept for every $x \in L \cup \overline{L} = \{0,1\}^*$.

We note that such proof systems wherein the prover can convince the verifier to accept for any element $x \in \{0,1\}^*$ include a trivial protocol such that the verifier always accepts without depending the prover's answer (e.g. the verifier simply says "OK").

To avoid such trivial protocols, we request another condition that the prover cannot convince the verifier without power to solve the problem. So, we consider the exact state of provers when the verifier accepts, and give a formal definition that *the prover has a program to compute a given function* by extending the notion of "possession of knowledge" defined in [FSS87] into "possession of programs which is correctly working [BK89]. "

2.2 Interactive proofs and program checkers

Interactive proofs, which were introduced by Goldwasser et al. [GMR85], is defined as follows.

Definition 2.1 [GMR85]: *Let (P,V) be a pair of probabilistic interactive TMs. We say that (P,V) is an interactive proof for membership of a language L if V is probabilistic, polynomial-time, and*

1. *For every $x \in L$, V accepts in its interaction with P on common input x with overwhelming probability.*
2. *For every $x \notin L$, and for every function P^* V accepts in its interaction with P^* on common input x with negligible probability.*

Note that the probabilities are over the random choice of both parties in the communication.

Beigel et al. [BBFG91] introduced the following variant in order to discuss how efficient the prover P can be in an interactive proof for membership of languages.

Definition 2.2 [BBFG91]: *Let P be a probabilistic polynomial-time interactive oracle TM and V be a probabilistic polynomial-time oracle TM. We say that (P,V) is a* `competitive` *interactive proof for membership of a language L if the following two conditions hold.*

1. *For every $x \in L$, V accepts in its interaction with P^L on common input x with overwhelming probability.*

2. *For every $x \notin L$, for every function P^*, V accepts in its interaction with P^* on common input x with negligible probability.*

Before introducing our new model, we recall the notion of program checkers defined by Blum and Kannan [BK89] as follows.

Definition 2.3 [BK89]: *Let C be a probabilistic polynomial time oracle TM. We say that C a checker for a function $F : \{0,1\}^* \to \{0,1\}^*$ if for all program P and all $x \in \{0,1\}^*$ it is the case that*

1. *If $P(y) = F(y)$ for all $y \in \{0,1\}^*$, then $C^P(x)$ accepts with overwhelming probability.*

2. *If $P(x) \neq F(x)$, then $C^P(x)$ accepts with negligible probability.*

Note that the probabilities are over the random coin tosses of the checker C.

2.3 The proposed model: demonstration Systems

Now we give the following definition of our model of demonstrating the correctness of programs for a function.

Definition 2.4: *Let P be a probabilistic polynomial-time interactive oracle TM and V be a probabilistic polynomial-time interactive TM. We say that (P, V) is a program-demonstration for a function $F : \{0,1\}^* \to \{0,1\}^*$ if the following two conditions hold.*

1. *For every $x \in \{0,1\}^*$, V accepts in its interaction with P^F on common input x with overwhelming probability, where P^F is a probabilistic polynomial time oracle TM with an oracle to compute the function F.*

2. *For any $x \in \{0,1\}^*$ and for any P^*, P^* can convince V to accept only if P^* actually computes the correct value of $F(x)$. A probabilistic polynomial-time oracle TM E is used in order to demonstrate P^*'s power to compute F. Formally:*

$$\forall a\ \exists E\ \forall P^*\ \forall x\ \forall b\ \exists c\ \forall |x| > c$$
$$Prob\Big(P^* \leftrightarrow V(x)\ \text{accepts}\Big) > 1/|x|^a$$
$$\Longrightarrow \quad Prob\Big(E^{P^*}(x) = F(x)\Big) > 1 - 1/|x|^b.$$

Note that the probabilities above is taken over all of the possible coin tosses of E and V.

Remark 2.5: Yung [Yun89] already formulated a notion of interactive proofs of computational power, which has a very similar goal to ours. Unlike our definition, Yung's formulation is not based on Blum's checker but founded on the protocols of zero-knowledge proofs of knowledge [FSS87, TW87] rather than the concept itself in [FSS87, TW87]. Therefore, Yung considered only computational power to solve problems which is restricted within (probabilistic polynomial-time) "samplable" and "verifiable" problems. We shall note that our definition based on Blum's result checking is a fundamental approach to deal with a wider class of ability to solve problems beyond NP.

Remark 2.6: Bellare and Goldreich [BG92] criticised previous definitions of proofs of knowledge [FSS87, TW87], and claimed that we have to deal with provers who convince the verifier with probability which is not non-negligible for wider applications. Indeed we shall adopt precisely the idea of Bellare and Goldreich [BG92], however, in this paper, we consider only provers who convince the verifier with non-negligible probability to clarify our idea of definition and simplify the discussion.

Zero-knowledgeness of program-demonstration systems is defined as one of interactive proof system for membership. We first recall that the *view* of the verifier is everything he sees during an interaction with the prover, that is, his own coin tosses and the conversation between himself and the prover.

Definition 2.7 [GMR85]: *Let (P,V') be an interactive protocol and let $x \in \{0,1\}^*$. The view of V' on input x is the probability space*

$$VIEW_{(P,V')}(x) = \{(R,C) : R \leftarrow \{0,1\}^{p(|x|)} ;\ C \leftarrow (P \leftrightarrow V'[R])(x)\},$$

where p is a polynomial bounding the running time of V', and $(P \leftarrow V'[R])(x)$ denotes the probability space of conversations between P and $V'[R]$ on input x (the probability is taken over the all of the possible coin tosses of P).

In the case of program-demonstration for the characteristic function of a language L, the condition of zero-knowledgeness requires that the simulator must exists for both $x \in L$ and $x \notin L$ (i.e. for any input $x \in \{0,1\}^*$), whereas zero-knowledge proofs for membership of language L discusses the simulator only for $x \in L$.

Definition 2.8: *A program-demonstration (P,V) for a function F is perfect zero-knowledge if there exists a simulator S which runs in expected polynomial time, for every V' and for $\forall x \in \{0,1\}^*$ $S(x;V'(x)) = VIEW_{(P,V')}(x)$.*

Intuitively, the definition above means that the verifier cannot get any information (even the value of $F(x)$) in its interaction with the prover.

3 Concluding remarks

We do not know if the obtained upper bound in Theorem II is tight. While an evidence for that SAT cannot have perfect zero-knowledge interactive proofs for membership is given in [For87], it remains open if the characteristic function of NP-complete languages (e.g. SAT) has a perfect zero-knowledge program-demonstration systems We shall note that even the problem whether SAT has a checker or not is still unsettled [BK89].

Another fundamental problem is to give a nice characterization of the class of the functions that have program-demonstration systems. It is already shown that the languages that have interactive proofs for membership (namely IP) coincides with PSPACE [LFKN90, Sha90]. Also Blum and Kannan [BK89] introduced the following variant class of IP called "function-restricted IP (fr-IP)," which is related to checkable problems. The set of all decision problems π for which there is an interactive proof system for Yes-instances of π satisfying the conditions that the honest prover must compute the function π and any prover (even dishonest one) must be a function from the set of instances to {Yes,No}. This fr-IP is shown to be equivalent to multi-prover interactive proof systems in which the honest provers can only answer membership of the language that they are being asked to prove [FRS88, BFL90].

Let CIP be the class of the languages that have competitive interactive proof systems, namely in which the honest provers can only answer menbership of the language that they are being asked to prove. It is easily observed that, for any $L \in$ CIP$\cap$ co-CIP, the characteristic function of L has a program-demonstrating system. An interesting problem is if the converse of this proposition holds or there exists a language that does not belong to CIP$\cap$ co-CIP, of which characteristic function has a program-demonstration.

References

[AH87] Aiello,W., and Hastad,J., "Statistical zero-knowledge languages can be recognized in two rounds," *JCSS*,vol.42,(1991); preliminary version in FOCS'87.

[BBFG91] Beigel, R., Bellare, M., Feigenbaum, J., and Goldwasser, S., "Languages that are easier than their proofs," FOCS'91.

[BCC88] Brassard, G., Chaum, D., and Crépeau, C., "Minimum Disclosure Proofs of Knowledge," JCSS. 37, No.2, (1988).

[BCY89] Brassard, G., Crépeau, C., and Yung, M., "Constant-round perfect zero-knowledge computationally convincing protocols," TCS, 84, (1991): preliminary version in ICALP'87.

[BFL90] Babai,L, Fortnow,L, and Lund,C., "Non-deterministic exponential time has two-prover interactive protocols," FOCS'90.

[BG92] Bellare, M., and Goldreich,O., "On defining Proofs of Knowledge," CRYPTO'92.

[BeGS94] Bellare, M., and Goldwasser,S., "The complexity of decision versus search," in SIAM J. Comp. (1994).

[BGG+88] Ben-Or, M., Goldreich,O., Goldwasser,S., Hastad,J., Kilian,J., Micali,S., and Rogaway.P., "Everything provable in provable in zero-knowledge," CRYPTO'88.

[BK89] Blum,M and Kannan,S., "Designing program that check their work," STOC'89.

[BMO90a] Bellare, M., Micali, S., and Ostrovsky, R., "Perfect Zero-Knowledge in Constant Rounds," Proc. of STOC'90,

[BMO90b] Bellare, M., Micali, S., and Ostrovsky, R., "The (true) complexity of statistical zero-knowledge," Proc. of STOC'90,

[BP92] Bellare, M., and Petrank, E., "Making zero-knowledge provers efficient," STOC'92.

[For87] Fortnow, L., "The Complexity of Perfect Zero-Knowledge," STOC'87.

[FSS87] Feige, U., Fiat, A., and Shamir, A., "Zero-Knowledge Proofs of Identity," J. of Cryptology, Vol.1, (1988); preliminary version in STOC'87.

[FS89] Feige, U. and Shamir, A., "Zero-Knowledge Proofs of Knowledge in Two Rounds," CRYPTO'89.

[FS90] Feige, U. and Shamir, A., "Witness Indistinguishable and Witness Hiding Protocols," STOC'90.

[GHY85] Galil,Z., Haber,S., and Yung,M., "Minimum-knowledge interctive proofs for decision problems," SIAM J. of Comput., Vol.18, No.4, (1989): preliminary version in FOCS'85.

[GK90] Goldreich, O. and Krawczyk, H., "On the Composition of Zero-Knowledge Proof Systems," Proc. of ICALP'90.

[GMR85] Goldwasser, S., Micali, S., and Rackoff, C., "The Knowledge Complexity of Interactive Proof Systems," SIAM J. of Comput., Vol.18, No.1, (1989): preliminary version in STOC'85.

[GMW86] Goldreich, O., Micali, S., and Wigderson, A., "Proofs that Yield Nothing But Their Validity or All Languages in NP Have Zero-Knowledge Proofs," FOCS'86.

[IY87] Impagliazzo, R., and Yung, M., "Direct minimum-knowledge computations," CRYPTO'87.

[JVV86] Jerrum,M., Valiant,L, and Vazirani,V., "Random generation of combinatorial structures from a uniform distribution," TCS vol. 43.

[KST93] Köbler,J., Schöning,U., and Tóran, J., "The graph isomorphism problem: Its structural complexity," Progress in TCS, *Birkhäuser*,(1993).

[FRS88] Fortnow,L., Rompel,J., and Sipser.M., "On the power of multiple-prover interactive protocols," Structure'88.

[Lau82] Lautemann,C., "BPP and polynomial hierachy," IPL vol.17.

[LFKN90] Lund,C., Fortnow,L, Karloff,H., and Nisan,N., "Algebraic methods for interactive proof systems," FOCS'90.

[Mat79] Mathon,R., "A note on the graph isomorphism counting problem," IPL, Vol.8 (1979).

[Sha90] Shamir,A., "IP = PSPACE," FOCS'90.

[Sip83] Sipser,M. "A complexity-theoretic approach to randomness," STOC'83.

[Tod89] Toda, S., "On the computational power of PP and $\oplus$P," FOCS'89.

[TW87] Tompa, M. and Woll, H., "Random Self-Reducibility and Zero-Knowledge Interactive Proofs of Possession of Information," FOCS'87.

[Yun89] Yung, M. "Zero-knowledge proofs of computational power," Proc. of Eurocrypt'89.